Mstánford
美斯坦福

SCCE软件工程师

使用ADO.NET开发三层架构应用程序

美斯坦福（中国）IT教育　编著

［第二阶段］

中国地质大学出版社
ZHONGGUO DIZHI DAXUE CHUBANSHE

美斯坦福(中国)IT教育授权合作院校指定教材

SCCE软件工程师

使用ADO.NET开发三层架构应用程序

美斯坦福(中国)IT教育　编著

◎第二阶段

中国地质大学出版社
ZHONGGUO DIZHI DAXUE CHUBANSHE

图书在版编目(CIP)数据

SCCE 软件工程师 / 美斯坦福(中国)IT 教育编著. — 武汉:中国地质大学出版社,2010.1

ISBN 978-7-5625-2453-3

Ⅰ. S…

Ⅱ. 美…

Ⅲ. 软件开发-工程技术人员-基本知识

Ⅳ. TP311.52

中国版本图书馆 CIP 数据核字(2010)第 011075 号

SCCE 软件工程师 **美斯坦福(中国)IT 教育 编著**

责任编辑:姜梅 责任校对:张咏梅

出版发行:中国地质大学出版社(武汉市洪山区鲁磨路 388 号) 邮编:430074

电话:(027)67883511 87395799 传真:67883580 E-mail:cbb@cug.edu.cn

http://www.cugp.cn

开本:880 mm × 1 230 mm 1/16 字数:4 350 千字 印张:137.125

版次:2010 年 1 月第 1 版 印次:2010 年 1 月第 1 次印刷

印刷:武汉市福成启铭彩色印刷包装有限公司 印数:1-6 000 册

ISBN 978-7-5625-2453-3 全套定价:550.00 元

前言 SCCE2010

欢迎学习 SCCE 国际软件工程师系列课程，我们致力于为 IT 企业培养拥有扎实的理论知识、丰富的项目经验、过硬的职业素质的复合型软件工程师。通过 SCCE 软件工程师系列课程，帮助学员提升核心竞争力，增加就业筹码、提升就业质量，获得可持续发展的信心和能力，早日实现人生价值。

信息技术产业（IT）作为发展最快的朝阳产业，是中国经济发展的重要内容。软件技术作为信息技术的核心，其发展速度和水平已然成为衡量一个国家现代化程度和综合国力的重要标志。过去的 10 年间，中国软件开发技术取得了长足的进步和发展，但与美国、日本、印度等 IT 强国相比仍有很大距离。为尽快缩短差距，中国政府出台了一系列的鼓励政策及措施，此时中国的 IT 产业正面临最好的发展机遇。

在 IT 行业高速发展的背后，IT 企业对于其核心竞争力—— IT 人才的渴求极其明显，招聘条件更趋于理性，要求也更为务实。IT 招聘中看重的不再是从业人员耀眼的教育背景、深奥的理论知识，而是转向能否“即插即用”、开发出有价值的项目；不光要求个人技术能力，更加注重职业素质；不光要求已有的知识体系，更加注重发展潜力。简而言之，IT 企业急需理论扎实、经验丰富、素质过硬的高端化、复合型人才。

美斯坦福（Mstanford）作为全球新晋的 IT 教育机构之一，拥有丰富的教育经验和国际领先的产品服务体系。自进入中国以来，依托其全球领先的教学模式、高质量的管理体系和与时俱进的课程内容，完善的就业安置体系，迅速成为众多有志青年进入 IT 产业的快速通道。本套课程即是美斯坦福（Mstanford）汇聚近百位权威专家，历时几百个日夜开发而成的最新力作。

教材研发过程中充分考虑到学员现有的知识体系及学习能力、采取以项目案例和知识模块为主线的“双核”内容组织模式，是本套课程第一大特色；以职业规划为主线，以 6 种能力、6 种心态、6 项准则为核心，以 10 大热门行业为背景的职业素质课程是本套课程的第二大特色；以企业需求为导向、以项目经验为突破口，通过 10 大行业的 36 个真实项目，共计 72000 行代码，积累相当于工作两年的项目经验是本套课程的第三大特色。以上三大特色使得本课程真正实现校企融合、以训带学，使学员实现从准职业人 → 职业人 → 成功企业人的快速转变和突破。

特别感谢研发团队每一位成员付出的辛勤劳动，也感谢给予研发团队支持和帮助的所有人！

祝所有学员学习顺利、学业有成！

课程说明 SCCE2010

一、为什么要使用三层架构

在 C#&WinForm 一书中，我们已经学会了 C# 的面向对象编程和 WinForm 应用程序的开发。但随着项目需求的增加，项目也将越来越大，需要多人分工来完成；项目维护也将越来越复杂，将业务逻辑与用户界面混编在一起不利于分工协作和代码的维护。这样就需要将一个大项目分成多个小项目或小模块，各模块之间分工明确，又可以互相协作来完成项目需求。开发人员可以将应用的商业逻辑放在中间层应用服务器上，把应用的业务逻辑与用户界面分开。在保证客户端功能的前提下，为用户提供一个简洁的界面。这意味着如果需要修改应用程序代码，只需要对中间层应用服务器进行修改，而无须修改成千上万的客户端应用程序，从而使开发人员可以专注于应用系统核心业务逻辑的分析、设计和开发，简化了应用系统的开发、更新和升级工作。所以，虽然新技术层出不穷、涌现出各种各样的架构模式，但三层架构自诞生至今在.NET 世界中始终保持有巨大的影响力。

二、课程内容模块介绍

第 1~3 章：介绍.NET 三层架构的基本原理及如何创建三层架构的系统；如何使用实体类对象在三层结构间传递数据；如何创建高效率的数据访问层和业务逻辑层。

第 4 章：介绍特性、序列化和反序列化、反射的概念及使用。学会自定义特性和使用反射。在本章的最后还讨论如何创建、安装和监视 Windows 服务。

第 5 章：介绍异常处理和测试。重点讨论如何使用 VSTS 创建、运行和观察单元测试。

三、课程内容学习目标

本课程学习完毕后，能够完成以下需求：

(1) 使用 ADO.NET 和三层架构开发高效的企业级应用程序。

(2) 能够使用序列化保存对象状态，使用反序列化构造对象。

(3) 能够在程序中应用特性标识编程元素。

(4) 能够利用反射来动态加载程序集并调用程序集中类的方法。

(5) 能够创建和监视 Windows 服务。

(6) 能够使用VSTS 工具对三层架构体系的应用程序进行单元测试。

目录

SCCE2010

第一部分 理论

Mstánford
美斯坦福

第一部分 理论

第1章

SCCE2010

企业级开发中搭建三层架构体系

本章学习内容

1. 三层架构体系的概念及优点
2. 三层架构中各层之间的逻辑关系和各自功能
3. 使用三层架构搭建项目框架
4. 使用 DataSet 作为业务实体

本章学习目标

1. 能够使用三层架构搭建项目框架
2. 能够使用 DataSet 在三层架构间传递数据

通过《使用 C# 开发.NET 应用程序》一书的学习，我们已经能够使用 C# 语言在.NET 平台上开发 Windows 应用程序。但至今为止，我们开发的应用程序都是将业务逻辑和用户界面放在一起。随着需求的增加，项目规模将越来越大，开发时需要多人分工来完成，后期维护也将越来越复杂，此时业务逻辑与用户界面混编不利于分工协作和代码的维护。需要将一个大项目分成多个小项目或小模块，各模块之间具有明确的分工，通过互相协作来完成项目需求。开发人员可以将应用的商业逻辑放在中间层应用服务器上，把应用的业务逻辑与用户界面分开。在保证客户端功能的前提下，为用户提供一个简洁的界面。三层架构能够实现这样的分工协作，它将应用程序分为表现层、业务逻辑层和数据访问层，各层之间协作实现应用程序的功能，从而分离用户界面和业务逻辑。表现层依赖业务逻辑层，业务逻辑层依赖数据访问层，使用数据对象（如 DataSet）在三层间传递数据。本章要求掌握使用三层架构搭建项目框架，并使用 DataSet 作为数据对象在三层之间传递数据。

1.1　三层架构概述

1.1.1　为什么需要三层架构

以现实生活中的肉类食品加工厂为例：一个大型的食品加工企业包括以下 3 个部分——屠宰场、食品加工厂、商场，如图 1.1.1 所示。

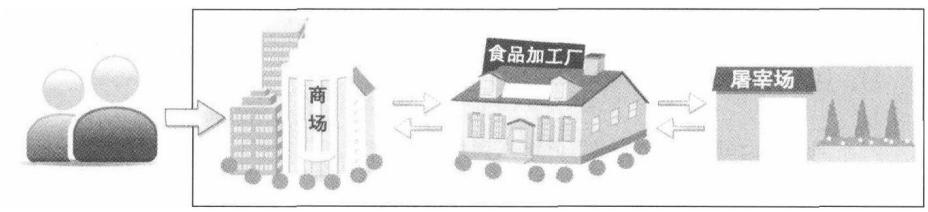

图 1.1.1　三层架构的食品加工厂

该业务模式下各部分的功能为：

(1) 顾客到商场购买肉食品。

(2) 商场负责接待购买肉食品的顾客。

(3) 商场从食品加工厂批量购入肉食品。

(4) 食品加工厂为商场提供肉食品。

(5) 食品加工厂从屠宰场获取原材料。

(6) 屠宰场负责提供原材料给食品加工厂。

食品加工企业将整个业务分为 3 部分来实现，这样做的好处是：其中一个环节发生变化时，不会影响到整个企业的业务，只需稍微改变其他环节即可。

来看看软件系统的情况，以一个登录模块为例，登录界面如图 1.1.2 所示。

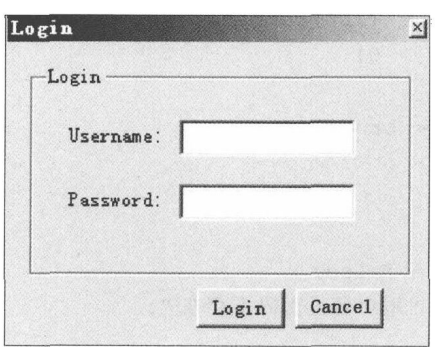

图 1.1.2　登录系统界面

示例 1.1

```
// 验证用户是否输入
private bool InputValidate()
{
    if (txtUsername.Text.Trim() == "")
    {
        MessageBox.Show("请输入用户名！");
        txtUsername.Focus();
        return false;
    }
    if (txtPassword.Text.Trim() == "")
    {
        MessageBox.Show("请输入密码！");
        txtPassword.Focus();
        return false;
    }
    return true;
}
// 验证用户名和密码是否正确
private bool UserValidate(string uid, string pwd,ref string msg)
{
    bool bFlag = false;
    string sql = string.Format("SELECT COUNT(*)
    FROM Users WHERE Username='{0}'and Password='{1}'", uid, pwd);
    SqlConnection connection = new SqlConnection
    ("server=.;database=RssDB;uid=sa; pwd=180705");
    SqlCommand command = new SqlCommand(sql, connection);
    try
    {
        if (connection.State == ConnectionState.Closed)
        {
            connection.Open();
        }
        int result = (int)command.ExecuteScalar();
        if (result > 0)
        {
            bFlag = true;
        }
        else
        {
            bFlag = false;
            msg = "用户名或密码不存在";
        }
    }
    catch (SqlException ex)
    {
        msg = ex.Message;
        bFlag = false;
    }
```

```
        finally
        {
            connection.Close();
        }
        return bFlag;
    }
    //用户登录
    private void btnLogin_Click(object sender, EventArgse)
    {
        if (InputValidate())//输入验证通过
        {
            string message = "";
            if (UserValidate(txtUsername.Text.Trim(), txtPassword.Text.Trim(),
            ref message))
            {
                //登录成功
            }
            else
            {
                //登录失败
                MessageBox.Show(message);
            }
        }
    }
```

上述代码可实现输入正确用户名和密码后进行登录的功能。但是，在程序完成以后，客户的需求有所改变，需要增加登录身份的验证，如图 1.1.3 所示。

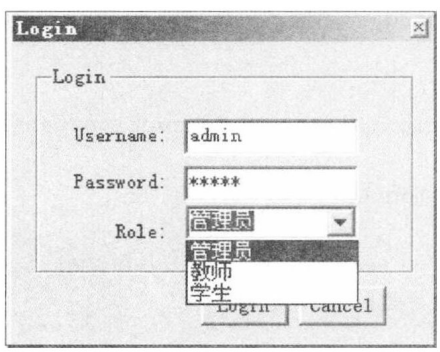

图 1.1.3　加入了身份验证的用户登录窗体

在这种情况下，不仅界面改变了，而且类代码也要做相应的改变，修改后的部分代码如示例 1.2。

示例 1.2

```
    //验证用户是否输入
    private bool InputValidate()
    {
        if (txtUsername.Text.Trim() == "")
```

```
        {
            MessageBox.Show("请输入用户名!");
            txtUsername.Focus();
            return false;
        }
        if (txtPassword.Text.Trim() == "")
        {
            MessageBox.Show("请输入密码!");
            txtPassword.Focus();
            return false;
        }
        //修改后的代码，添加了身份验证功能
        if (cboRole.Text.Trim() == "")
        {
            MessageBox.Show("请选择登录身份!");
            cboRole.Focus();
            return false;
        }
            return true;
    }
    //验证用户名和密码是否正确
    private bool UserValidate(string uid, string pwd,string role,ref string msg)
    {
        bool bFlag = false;
        //查询语句增加了一个条件
        string sql = string.Format("SELECT COUNT(*) FROM Users WHERE
        Username='{0}'and Password={1}' and Role='{2}'", uid, pwd,role);
        SqlConnection connection = new SqlConnection("server=.;database=RssDB;
        uid=sa;pwd=180705");
        SqlCommand command = new SqlCommand(sql, connection);
        try
        {
            if (connection.State == ConnectionState.Closed)
            {
                connection.Open();
            }
            int result = (int)command.ExecuteScalar();
            if (result > 0)
            {
                bFlag = true;
            }
            else
            {
                bFlag = false;
                msg = "用户名或密码不存在";
            }
        }
        catch (SqlException ex)
        {
            msg = ex.Message;
            bFlag = false;
        }
```